Geo-referenced Photo Management System Context-Aware Systems 2022/2023 Report

Giuseppe Massimo
Master's degree student
University of Bologna
Email: giuseppe.massimo@studio.unibo.it

Stefano Notari
Master's degree student
University of Bologna
Email: stefano.notari2@studio.unibo.it

I. Introduction

With this demo, we want to demonstrate how a system for managing geo-referenced photos works. The project consists of a backend that provides APIs to load photos or collections into the database, a Mobile Client which is an Android application that allows users to take photos, upload them with spatial geo-tags, and view the list of N photos closest to the user's position. Furthermore, a Webapp for the frontend has been developed, which provides a web dashboard for performing various operations, such as displaying uploaded photos on the map using markers and implementing some features related to the spatial geo-tag. These features include the ability to upload a GeoJSON file to create geographic borders, visualize a Heatmap to show the intensity of markers in a specific zone, and the implementation of some clustering algorithms (K-means, DBScan).

August 1, 2023

II. Project's Architecture

The project is composed by the following components:

- frontend (webapp): dashboard for visualizing the positions of uploaded photos on a map and implementing features based on the photo's geo-tag
- app (Android): an application that allows users to take photos and upload them with the geo-tag taken from the user's current position. It also shows photos nearest to the user's location
- backend: an application that implements and shares the API for managing collection and photo resources. It also handles metadata and geo-tags for the photos' positions
- postgresql [6] (with postgis extension): database where the backend stores the metadata of photo/collection and the geo-tag of photos

The Figure 1 shows a high level schema of implemented components and their interaction. In this case the deploy is done on the local computer without the containers and kubernetes.
The local deploy is useful during the development of the application because it allows to have the code automatically recompiled when updated and to ignore the complications of kubernetes.

Once the application was in a stable version, that version was tested through the deployment in kubernetes using minikube [2].
Minukube is a tool that allows to run a single-node Kubernetes cluster locally. The Figure 2 shows a high level description of this scenario.

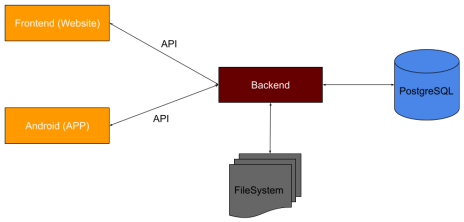

Fig. 1. Architecture of the project

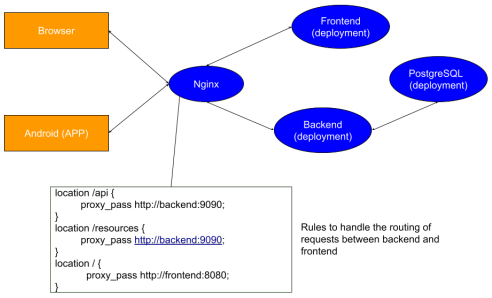

Fig. 2. Architecture of the project with kubernetes

A. Database seeding

To test the features of both the app and frontend, it was necessary to have some data inserted into the database. This operation can be time-consuming and error-prone, so it was decided to automate this task.

Specifically, each time the backend starts, it executes one query for the collection table and one for the photo table. If there aren't any records, it proceeds to execute a list of inserts with the data to initialize those tables.
To make this solution work, it is required to add a line in the Dockerfile of the backend to copy the photos from the local storage to the container.

III. Project's Implementation

The components illustrated in Section II have been implemented using the following technologies:

- the frontend application has been developed with the Vue.js framework. [8]
- the mobile application has been implemented with Java
- the backend has been implemented with Node.js [5] and TypeScript [7]

To handle the communication between the backend and the database, the pg library has been used. This client allows for handling the communication with the database and running queries.
Both the frontend and the mobile app send REST requests to the backend using the APIs implemented with the Express framework [1].

A. Kubernetes and Nginx

When the application is deployed with Kubernetes, the containers are in distinct pods but on the same node, so we have that both the requests for the frontend and the backend are on the same port.
So, in order to distinguish the requests for the frontend application from the requests of the backend, it is necessary to add Nginx [4].
Nginx is a web server that allows writing rules that redirect the requests to a new host (container) following the pattern matching of the PATH. The rules used in our application are reported in Figure 2.

B. Upload of files - Multer

To handle the upload of photos has been used the multer library [3].
All the photos uploaded are stored in the same directory and to make each filename unique is added the timestamp before the file's extension.

IV. Results

In this section, the obtained results will be shown, displaying the interface of both the frontend and the mobile app.

A. Frontend

1) Create new collection: In the figures 3 and 4 is it possible to visualize the operation of the "Upload" section, where you can create a new collection and add photos to an existing collection

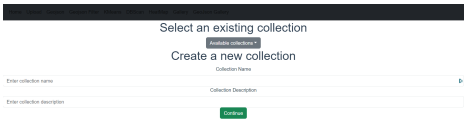

Fig. 3. Create new collection

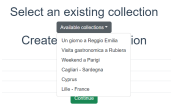

Fig. 4. Select existing collection

2) Upload GEOJSON file: In the GeoJSON section is possible to upload a JSON file with geo-spatial data and remove those already uploaded like is possible to see in figures 5 and 6

Fig. 5. Upload GeoJSON File

Fig. 6. Remove GeoJSON

3) Marker and filters view: In the GeoJSON Filter is possible to see the list of geografical borders contained in JSON file.

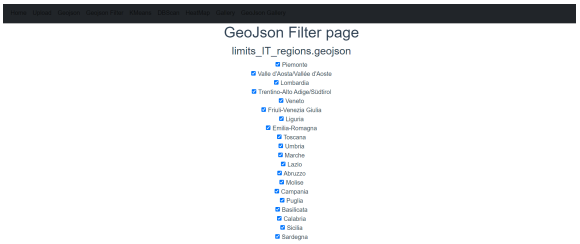

Fig. 7. GeoJSON Filter

4) Map coloring: After selecting the filters on the Homepage, in addition to the marker display, it is possible to view the map with the selected boundaries, and each zone is colored from blue to red based on the number of photos (markers) present within that boundary.

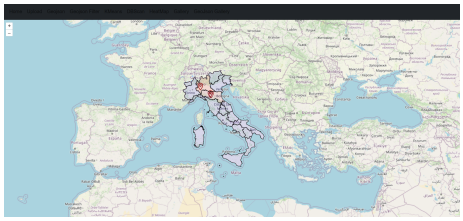

Fig. 8. Map Color

5) Clustering: In this section, it will be shown the results of clustering algorithm.

- K-means [10]: In the Figure 9, it is possible to see the results of kmeans algorithm set up with 5 cluster.

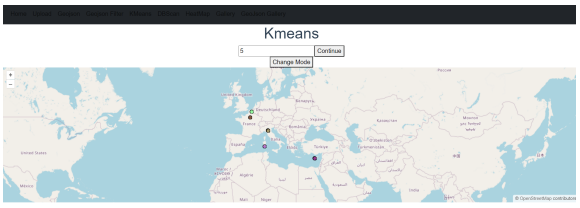

Fig. 9. K-means

- K-means (elbow): In the figure 10, it's shown a version of k-means where the number of cluster are not set by user but automatically handled by searching the cluster with the minimum SSE.

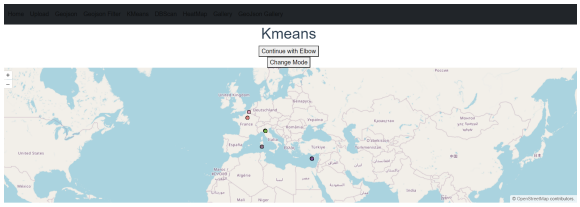

Fig. 10. Elbow

- DBSCAN [9]: As an optional feature, an alternative algorithm of clustering, the DBSCAN, has been implemented. The DBSCAN algorithm doesn't require the number of clusters but only the radius and the minimum number of elements to create a new cluster.

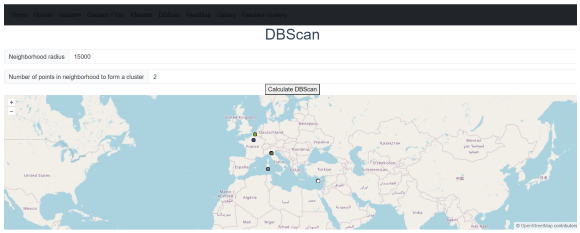

Fig. 11. DBScan

6) Heatmap: In the Heatmap section, it is possible to visualize on the map, in the form of points, the intensity of markers present in a specific zone.

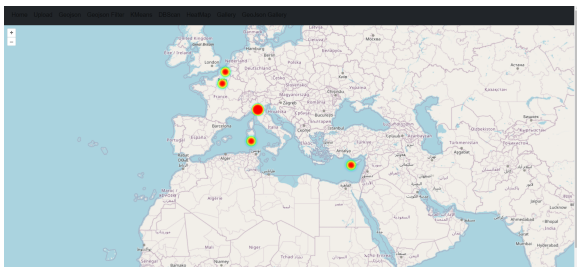

Fig. 12. Heatmap

7) Gallery: In the ”Gallery” section is possible to select a collection and view the photos contained in it.By going to the ”GeoJSON Gallery” section, it is also possible to view the photos inside a boundary outlined by the GeoJSON file.

Fig. 13. Gallery

Fig. 14. Gallery view

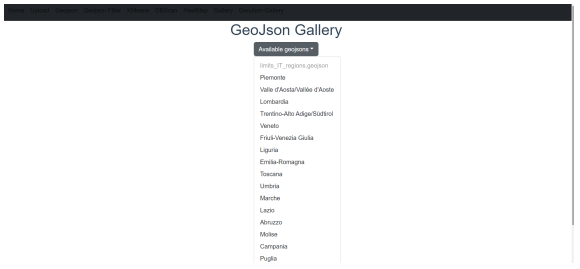

Fig. 15. GeoJSON Gallery

Fig. 16. GeoJSON Gallery view

B. App

1) Home: In the figure 17 is possible to see the home screen of the Android app with its respective Navigation Drawer.

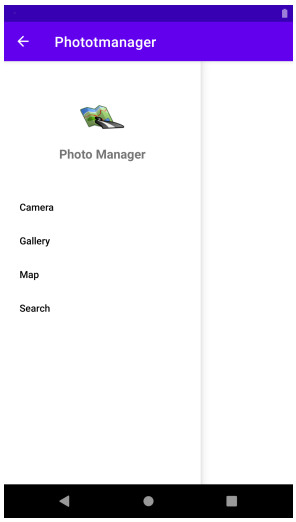

Fig. 17. Navigation Drawer

2) Camera: Once selected from the Navigation Drawer, the Camera will be available to take a photo, and you can decide whether to create a new collection or add the photo to an existing collection (Figure 18). In the case of creating a new collection, a name and description for the collection will be required (Figure 19). On the other hand, if the photo is to be added to an existing collection, a name for the photo and the selection of the collection where you want to add it will be required (Figure 20).

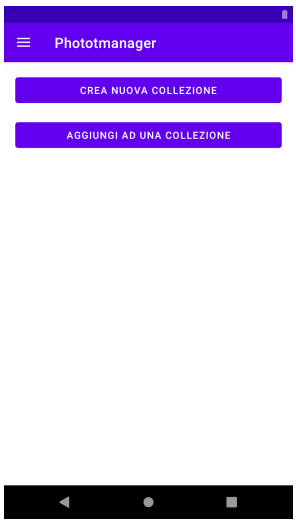

Fig. 18. New collection or Add to collection

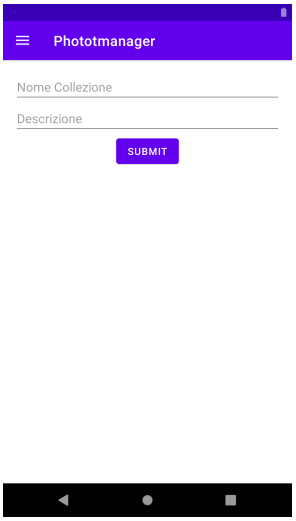

Fig. 19. New collection

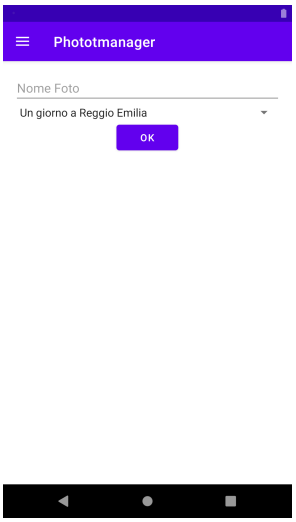

Fig. 20. Add to collection

3) Map: In the ”Map” section will be opened a map where it will be possible to visualize the markers at the points where the photos are taken.

Fig. 21. Map

4) Search: With the ”Search” feature is possible to insert a numeric value (Figure 23) that provide to visualize the N photos closest to user position (Figure 24).

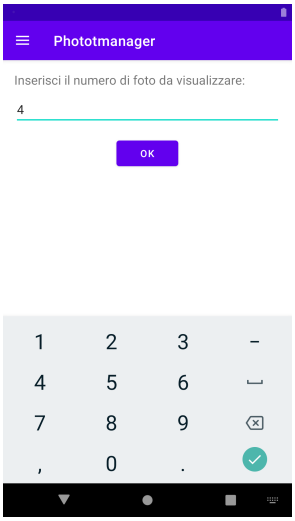

Fig. 22. Search nearest photos

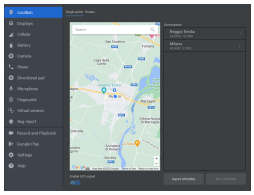

Fig. 23. Location

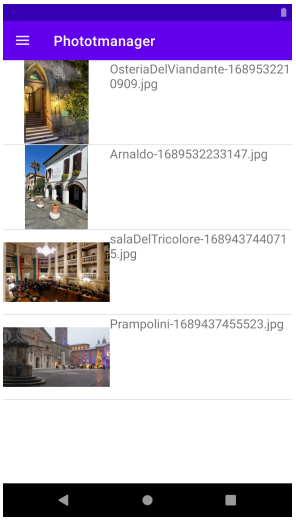

Fig. 24. N photos

5) Gallery: In "Gallery" section is possible to see all photos of a selected collection (Figure 25).

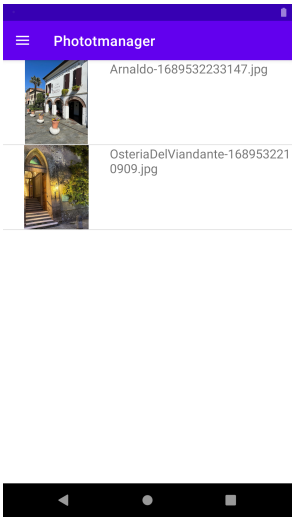

Fig. 25. Gallery

REFERENCES

[1] Express - Node.js web application framework. https://expressjs.com/. Version 5.0.4.

[2] Minikube. https://minikube.sigs.k8s.io/docs/. Version 1.13.0.

[3] Multer. https://github.com/expressjs/multer. Version 1.4.5.

[4] NGINX: Advanced Load Balancer, Web Server, Reverse Proxy. https://www.nginx.com/. Version 1.24.

[5] Node.js. https://nodejs.org/en/. Version 16.10.0.

[6] PostgreSQL. https://www.postgresql.org/. Version 15.0.0.

[7] Typescript. https://www.typescriptlang.org/. Version 5.0.4.

[8] Vue.js. https://vuejs.org/. Version 3.2.13.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[10] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.